Dynamical Climatology

The Met O 20 Stratosphere-Mesosphere Model

by

M. Fisher

DCTN 52

March 1987

Meteorological Office (Met. O. 20)
London Road
Bracknell
Berkshire RG12 2SZ

KW

DYNAMICAL CLIMATOLOGY TECHNICAL NOTE NO.52

THE

MET O 20

STRATOSPHERE - MESOSPHERE MODEL

by

M. FISHER

MET O 20 (Dynamical
Climatology Branch)
Meteorological Office
London Road
Bracknell
Berkshire RG12 2SZ

March 1987

Note: This paper has not been published. Permission to quote from it should be obtained from the Assistant Director of the above Meteorological Office Branch.

Contents

	Page
Introduction	4
Description of the model	5
Background to the FORTRAN-77 Model	9
Design Requirements for the FORTRAN-77 Model	11
Modular Structure	14
The Source Code Pre-processor	15
I/O Routines	16
The Paging Scheme	18
Time Filter	21
On the Use of an Increment Array	23
Restarting and Data Spooling	25
Description of a Model Run	28
The Main Arrays, Variables and Parameters	30
The Dynamics Scheme	36
Shapiro Filters	41
Data Requirements	43
Output from the Model	46
Conservation Properties	47
References	53
Appendix I Alphabetical List of Subroutines and Functions	56
Appendix II Dump Format Data	61
Appendix III Paging Scheme Performance	62
Appendix IV Notation	64
Appendix V Finite Difference Accuracy Near the Poles	65

Appendix VI	List of Constants in File PARAMS	67
Appendix VII	Transporting Data between Computers	69
Figures		71
Listing 1	The Base Code	Microfiche
Listing 2	Pre-processor Directives (IBM version)	"
Listing 3	Pre-processor Directives (Cyber version)	"
Listing 4	Transport Format Data Conversion Programs	"
Listing 5	Direct Data Conversion Programs (IBM/Cyber)	"

Introduction

This note describes the Met O 20 FORTRAN-77 stratosphere-mesosphere model: a portable, global, three dimensional, primitive equation model of the middle atmosphere.

The model exists in several versions, each of which may be implemented on more than one computer. Each implementation and version is produced by adding to and modifying a basic set of code called the base code. Apart from a few minor corrections, the unmodified base code is all that is required to run the simplest version of the model on the Met. Office's IBM computers. It is this version which is documented here. The modifications to run the model on a Cyber 205 computer are included on the microfiche which accompanies this note.

Description of the Model

The stratosphere-mesosphere model is a global, primitive equation model of the middle atmosphere. The equations are solved to fourth order accuracy in the horizontal and second order accuracy in the vertical and in time using energy conserving "box" type finite differences and leapfrog integration. Model fields (temperature and the horizontal components of wind) are held on an A-grid with grid points equally spaced in longitude, latitude and log pressure.

The model avoids having to represent the troposphere by imposing an artificial lower boundary condition near the tropopause. The geopotential height one grid space below the first level of the model is specified by a sequence of fields which are linearly interpolated in time to provide a continuous, time varying field.

The discrete equations of the model are given below. (The notation is defined in Appendix IV).

$$\delta_t \overline{u}^t + D(u) - (f + \frac{u}{a}\tan\phi)v + (a\cos\phi)^{-1} \left[\frac{4}{3}\delta_\lambda \overline{\Phi}^\lambda - \frac{1}{3}\delta_{2\lambda}\overline{\Phi}^{2\lambda} \right] = F_\lambda$$

$$\delta_t \overline{v}^t + D(v) + (f + \frac{u}{a}\tan\phi)u + (a\cos\phi)^{-1} \left[\frac{4}{3}[\delta_\phi(\overline{\Phi}^\phi\cos\phi) - \Phi\delta_\phi(\cos\phi)] - \frac{1}{3}[\delta_{2\phi}(\overline{\Phi}^{2\phi}\cos\phi) - \Phi\delta_{2\phi}(\cos\phi)] \right] = F_\phi$$

$$\delta_t \overline{T}^t + D(T) - \frac{\kappa(\overline{p}^s)T}{p} = Q/C_p$$

$$D(1) = 0$$

$$\delta_s \Phi + R\overline{T}^s = 0$$

where $s = \log(p/p_o)$

and

$$D(X) = (a\cos\phi)^{-1} \{ \frac{4}{3} [\delta_\lambda(\bar{u}^\lambda \bar{X}^\lambda) + \delta_\phi(\bar{v}^\phi \bar{X}^\phi \cos_\phi)]$$

$$- \frac{1}{3} [\delta_{2\lambda}(\bar{u}^{2\lambda} \bar{X}^{2\lambda}) + \delta_{2\phi}(\bar{v}^{2\phi} \bar{X}^{2\phi} \cos\phi)] \}$$

$$+ \frac{1}{p} \delta_s(p\dot{s}\, \bar{X}^s)$$

In addition to the geopotential height, solution of the equations requires knowledge of temperature and horizontal wind one grid space below the first level of the model. These are calculated by linear extrapolation from the first and second model levels. The upper boundary condition sets the "vertical velocity" $\dot{s}$ to zero half a grid space above the highest level of the model.

The latitudinal arrangement of grid points is such that the North and South poles are half a grid space from the nearest grid points. Near the poles, the finite differences require values of temperature and wind at grid points which are across the poles. The equations above are not correct in this case as they do not account for the change in direction of the northwards and eastwards unit vectors across the poles. To remedy this, cross-polar values of the wind components are multiplied by -1 before being used.

Note that although pressure appears in several terms in the equations, it invariably occurs as a ratio of pressure at neighbouring levels. Since the levels are equally spaced in $\log (p/p_o)$, the ratio is the same for each level. Consequently, the equations are identical at each level and may be expressed in terms of coefficients which are either constant or vary only with latitude. This eliminates the need for arrays of coefficients and allows efficient use to be made of vector computers such as the Cyber 205.

A problem with "box" type differences is the loss of accuracy near the poles of the approximation to the meridional gradient of geopotential (Kalney-Rivas 1978). Appendix V proves that this term is at worst third order accurate in the otherwise fourth order accurate equations above. The energy conserving property of the equations is proved in a later section.

Parametrizations of radiation and gravity wave induced drag are at present extremely simple, consisting of linear relaxation towards a prescribed zonal mean temperature field and zero wind. The coefficients used are those of Holton (1972b) and are given below. (See also Figs. 3 and 4).

temperature $\alpha(z) = \{1.5 + [(z-35)/7]\} \times 10^{-6}$ s^{-1}

wind $\alpha(z) = 10^{-7}$ s^{-1} $z<50$

$= 10^{-7} + 5 \times 10^{-6} \times (1 - \exp[(50-z)/40])$ s^{-1} $z\geq 50$

where z= height (km) above sea level.

Alternative physical parametrizations have been developed for the model by Drs Shine and Barker at the Department of Atmospheric Physics of the University of Oxford (Barker 1985, Shine 1984, 1985a, 1985b). However, these have yet to be implemented in the FORTRAN-77 model and are not included in the base code. They are therefore not documented in this note.

The convergence of meridians towards the poles of a regular latitude-longitude grid and the use of an explicit integration scheme impose severe restrictions on the maximum stable time step unless measures are taken to reduce the model's effective resolution near the poles. The stratosphere-mesosphere model achieves this by Fourier analysing the fields

near the poles and removing waves with high zonal wave number. The Fourier analysis is performed using one of the self-sorting, mixed radix FFT routines developed by Temperton (1976, 1979, 1982).

Williamson (1976) has tabulated the maximum stable time step as a function of resolution and Fourier truncation for the linear, shallow water equivalent of the discrete equations solved by the model. This provides a useful first guess of the degree of truncation required for a given timestep and resolution. The user is warned against excessive truncation as it is thought to have a detrimental effect on the model's ability to conserve potential vorticity in cross-polar flow regimes.

Non linear instability is prevented by the use of a 17 point filter (Shapiro 1970) which is applied to the model's fields in both the zonal and meridional directions after every time step. The response function of this filter is depicted in Fig. 5.

The leapfrog integration scheme requires the use of some method of suppressing computational modes which lead to separation of the time levels. This is achieved through the use of a mild time filter and by performing a forward timestep at intervals throughout the integration immediately after writing to disk. The latter also enables exact repetition of an integration from any output dump without needing to store both time levels.

Background to the FORTRAN-77 Model

Before 1982, the stratosphere-mesosphere model existed as a FORTRAN-66 program which was run on the Met Office's IBM 360/195 computer. This model had a resolution of 22.5° in the zonal direction which, due to a somewhat rigid data format and the lack of PARAMETER statements in FORTRAN-66, could not be altered without careful recoding and testing. The model, as formulated by Murgatroyd (1971), used second order accurate finite differences but was later modified to use fourth order accurate finite differences.

In 1981, the Met. Office acquired a Cyber 205 computer. However, mainly because of the indirect addressing which had been used to implement the fourth order differences, the model could not easily be modified to take advantage of the Cyber 205's vector processing capability. The model was therefore recoded by the author, and the opportunity was taken to introduce a more flexible data format and to parametrize the model's resolution.

The FORTRAN compiler for the Cyber 205 would not initially accept half precision arithmetic except through the use of "special calls". (These are machine instructions which can be inserted into a Cyber FORTRAN program by coding statements which look like subroutine calls). It was decided therefore, to code the model with special calls for all vector operations. This, of course, meant that the model could only be run on a Cyber 205.

The lack of portability of the "special calls" version of the model became a major problem during recent collaborative work with the Dept. of Atmospheric Physics at the University of Oxford. Part of this collaboration involved the development at Oxford of improved physical parametrizations for the model. As there was neither easy access to a

Cyber 205 from Oxford nor sufficient expertise in Cyber 205 machine language, the parametrizations were developed for the old (IBM) version of the model and have required substantial changes to be made to the code and lengthy testing before they could be used in the "special calls" version.

The main reason for producing a new, FORTRAN coded model was to allow the staff at Oxford and at the Met. Office to work with the same model program whilst retaining the speed of the vector code of the "special calls" model. There are, however, further advantages in using standard FORTRAN, some of which are listed below:

(i) The model is legible and can be understood by people who do not have Cyber 205 experience.

(ii) The model will not require re-writing when the Cyber 205 is replaced.

(iii) The model can be run on the CRAY-XMP computer at ECMWF, thereby using some of the Met. Office's large allocation of time on that machine.

(iv) Program development can be carried out on the Met. Office's IBM system, taking advantage of foreground editing and compilation and fast job turn around.

(v) It is hoped that a portable model will encourage future collaboration between the Met. Office and members of the scientific community with an interest in middle atmosphere modelling.

Design Requirements for the FORTRAN-77 Model

The model was designed to meet the following requirements:

1. The code should be portable.

Thus (a) The code should adhere strictly to the FORTRAN-77 standard (ANSI X3.9 - 1978).

(b) The model should not depend on the use of a particular operating system.

(c) The model should not depend on hardware features (e.g. word length) of a particular computer.

2. Modifications to the code should be portable.

3. Model data should be portable.

4. The code should be efficient on a vector computer.

5. It should be possible to run the model at very high resolutions which require more than the physical memory of a virtual storage machine.

6. The program architecture should allow easy adaptation to run efficiently on a multiprocessing computer.

7. The code should be modular, legible and easily understood and modified by its users.

These requirements place some constraints on the model program.

Portability of code means that I/O must be performed using standard READ and WRITE statements rather than efficient operating system calls or concurrent I/O. This is not a major problem, as even on a vector computer the model is CPU bound (CPU time is 93% of wall clock time on a Cyber 205).

Since FORTRAN-77 has no explicit vector syntax, the code must rely on automatic "vectorization" during compilation. The FORTRAN language is poorly suited to this process and "vectorizers" require a lot of help from the programmer. In particular the code must consist of simple, tightly

nested DO loops with constant loop bounds and the statements inside the loops must not allow the "vectorizer" the slightest chance of detecting a "possible vector dependency" (CDC compilers are especially fussy in this respect).

Modifications to the code are made portable through the use of a source-code pre-processor. All users have the same basic model code (called the base code) which is never modified directly. Instead, changes to the code are applied by the pre-processor which produces a file in a form suitable for compilation. Since all users have a copy of the pre-processor and apply their modifications to the same base code, these modifications are portable. Individual users can also extend their base code by adding new routines without destroying this portability.

Communication of data between users is performed by using a "transport format" (described in an appendix) in which data is stored as characters in fixed length records. FORTRAN-77 programs exist to convert data between "transport format" and the format required by the model.

In order to run the model at very high resolution a paging scheme must be used on present day computers. There are two possible approaches for such a scheme. Either the paging can be performed explicitly by READing and WRITEing data into and out of arrays which are smaller than the phyiscal memory of the machine, or the model can be written with arrays which may be larger than the machine's memory and the paging scheme simply informs the operating system which parts of the arrays are, or will shortly be, required for computation and which parts of the arrays can be paged to disk. This is called implicit paging. An explicit paging scheme is suitable for a model which is usually larger than the available memory. If the model is run at low resolution then the paging scheme must still

operate (although the READs and WRITEs may be replaced by copies to a buffer area in memory). An implicit paging scheme, on the other hand, is more suitable for a model which is usually smaller than the available memory (in which case the paging scheme may be completely removed). It does, however, have the disadvantage that calls to the operating system are required and can only be used on a virtual storage machine. As the paging scheme will not be required by most users, and may soon become redundant or impractical because of advances in computer technology, an implicit scheme was chosen for the stratosphere-mesosphere model.

The main architectural requirement of a paging model is that all calculations are performed on one part of the grid before proceeding to the next part. In the stratosphere-mesosphere model, this is achieved by performing the calculations latitude by latitude. This has the advantage that the calculations at each latitude may be performed independently of those at another latitude. On a multiprocessing computer, several latitudes may be processed by independent, concurrent tasks. The stratosphere-mesosphere model is well suited to this method of multiprocessing.

Modular Structure

The model is highly modular, consisting of nearly seventy subroutines, each of which may be regarded as performing a single function. Most of these subroutines are quite short and can be listed on a single side of paper. This improves legibility, facilitates testing, and in conjunction with a top-down programming method, resulted in a short development time for the code.

Fig. 1 illustrates the modular structure of the model. All the subroutines and functions which make up the model are included in Fig. 1 except for standard FORTRAN functions (e.g. LOG and SIN) and routines which are called by the utilities STOP, DATIM and FFT which were not written by the author. Subroutine or function calls are illustrated by linking calling routines on the left with called routines on the right. Arrows indicate that the diagram is continued elsewhere in Fig. 1.

One advantage of a modular approach to programming is the ease with which global modifications can be made to the code. For example, the model currently uses a real calendar to express time since it is used mainly to simulate real events. However, for climate studies, it is often more convenient to use a 360 day calendar in which each month contains 30 days. Because of the modular nature of the code, all manipulations of dates are carried out in three simple date handling routines. It is a trivial matter to change these routines to operate with a 360 day calendar, thereby changing every manipulation of dates in the model to the new calendar.

The Source Code Pre-processor

The use of a source code pre-processor is an important aspect of the flexibility and portability of the stratosphere-meshosphere model. It has two main functions. Firstly, the pre-processor is used to insert the PARAMETER statements which define the dimensions of the model's grid into all subroutines and functions of the model which use them. By copying the statements from a single set of code (called a common deck), the pre-processor allows the resolution of the model to be changed by modifying one line of code. Secondly, the pre-processor is used to make modifications to the code by inserting or deleting individual lines and by selecting or excluding whole subroutines. Because all modifications to the code are made using the pre-processor, the basic source code may be "frozen" so that, apart from the addition of new routines, all users work with the same code. This makes modifications to the code portable, allowing easy interchange of code between users.

Since the stratosphere-mesosphere model will generally be run on CDC or CRAY computers, the UPDATE utility which is available on these machines is the obvious choice for a source code pre-processor. However, UPDATE is not available on all machines, in particular the Met Office's IBM system. To avoid limiting the portability of the model, an UPDATE-like pre-processor has been written in FORTRAN-77. This program accepts an important subset of UPDATE directives, namely *DECK, *COMDECK, *CALL, *ID, *C, *I, *D and */. The program is considerably simpler than a full implementation of UPDATE and therefore has some limitations which should be recognised when transporting modifications between computer installations. The reader is referred to a separate note (Fisher 1986b) for further information.

I/O Routines

Initial, boundary and output data share a common format which is written and read by the model's I/O routines BUFIN and BUFOUT. Files are partitioned into dumps, each of which is valid for a particular time and date, and contains a header and a number of fields of data. A header consists of 1024 words containing the date of the dump, a title for the data and, in the case of an output dump, a copy of the model's IPARMS and RPARMS arrays.

Dumps can be accessed sequentially by calls to the I/O routines of the form:

CALL BUFIN (unit, array, command)

and CALL BUFOUT (unit, array, 1024, command)

where "unit" is the FORTRAN unit number of the file. "command" may be either "*TOP " or "*NEXT " and in either case causes a header to be transferred to or from "array".

Calls to BUFIN of the above form position the reading device at the first or next dump in the file. Fields can then be read from the dump as described below. Calls to BUFOUT of the above form start a new dump by writing its header (the header is always the first part of the dump to be written). If "*TOP " is specified then the file is rewound and the header is written to the start of the file.

Having transferred the header of a dump, fields of data may be written to or read from the file by calls to the I/O routines of the form:

CALL BUFIN (unit, array, name)

and CALL BUFOUT (unit, array, length, name)

where "length" specifies the number of elements of "array" to be written to the file and "name" is an eight character name which is associated with the data.

The fields of data within a dump may be read in any order by BUFIN. However, I/O efficiency is improved if the fields are read in the same order as they were written. Note that "length" is not required for a call to BUFIN as the number of elements of data transferred by such a call is always the same as the number of elements which were written by the corresponding call to BUFOUT.

The way in which dump format data is stored in a file need not concern most users of the model as they will always use BUFIN, BUFOUT or higher level routines to manipulate their data. For completeness, however, appendix II describes the method used.

The Paging Scheme

To allow the possibility of conducting extremely high resolution integrations, the model contains an implicit paging scheme. This permits the virtual memory required by the model to exceed the available real memory by guiding the operating system in its paging decisions. The advantages for the model of implicit paging over an explicit scheme are discussed in a previous section.

The paging scheme assumes the existence of a subroutine ADVISE whose arguments include an array reference and an integer 'n' (say). ADVISE informs the operating system that 'n' array elements are either no longer required or will shortly be required for use in the model's calculations. The operating system uses this information to decide which areas of virtual memory should be mapped onto real memory.

Whilst a non-paging model is CPU-bound even on a "super-computer", the situation is markedly different for a paging model. Since paging must occur at every time step, whereas a non-paging model need only perform I/O every model day, the increase in the amount of I/O required may be two orders of magnitude or more. Such a large increase in I/O activity makes the model I/O-bound. It is important, therefore to ensure that the amount of paging required is minimised, and that as much of the I/O as possible is concurrent with CPU activity.

Paging is reduced by making as much of the model as possible reside permanently in real memory. e.g. nearly half of a model which is twice as large as the available real memory may reside permanently in real memory. Paging and CPU activity are made concurrent by "advising" the operating system to page in areas of the model as far in advance of their anticipated use as possible. This is achieved by interspersing those areas of the

model which are subjected to paging at regular intervals between the areas which always reside in memory. Paging is thus concurrent with the calculations for the intervening, memory resident areas of the model.

The paging scheme in the base code model is simplified by grouping each latitude of the grid on a page boundary. The scheme can therefore page each latitude independently. This has the disadvantage on a Cyber 205 computer that paging must be performed in units of small pages. This results in a considerable loss of efficiency over large page I/O. However, only minor modifications to the code are necessary to change the scheme to use large page I/O.

Appendix III presents the calculations necessary to estimate the time required per timestep for various resolutions of the model.

Users of the model should be aware of two important factors before using the paging scheme.

Firstly, the base code model cannot be run with the paging scheme since the North-South Shapiro filtering is performed outside the model's loop over latitude. The filter must be removed and replaced by an alternative filter or diffusion scheme which is called by the model's UPDLAT routine.

Secondly, it is important not to overestimate the size of the available real memory. In particular, it should be noted that the size of memory given by the NSPCOR parameter in the TIMSTP routine should not include the space required by the model code, WORK array, I/O buffers, etc. Overestimating this parameter will result in paging taking place which has not been "advised" by the paging scheme. Most operating systems will page

out the least recently used area of memory unless advised otherwise. In the stratosphere mesosphere model, this is precisely the area of memory which is required next!

It should be noted that, as written, the paging scheme is of limited use on a Cyber 205 computer because the speed of small page I/O is only about 3% of the speed with which the model performs its calculations for any given area of memory. (This figure may be improved to about 30% if large page I/O is used). The model therefore becomes I/O bound if the model is more than 3% larger than available memory. The speed of the model is then determined by the speed and amount of I/O and decreases rapidly with increasing size of model. Despite this it is recommended that the programmer discipline needed to write code for a paging model is maintained in view of the ease with which such a model can be adapted to run on a multi-processing computer.

Time Filter

The computational modes which lead to divergence of the time levels of the leapfrog integration scheme are characterized by their alternate sign at consecutive time steps. It is desirable to remove or control the amplitude of these modes to prevent them from interacting, via the non-linearities of the system, with the physical modes of the model. One method of removing the computational modes is to perform a forward time step at intervals throughout the integration. However, this should not be done too frequently as it can lead to numerical instability (Hinds 1982). The stratosphere-mesosphere model performs a forward time step immediately after writing to disk. Between forward time steps, the amplitude of the computational modes is controlled by the use of a mild time filter. This also damps the high frequency gravity wave modes of the model.

Ideally, a non-recursive, symmetric filter should be used, as this would not alter the phases of wave components. The simplest such filter is

$$F : X(t) \rightarrow X(t) + (\nu/2)(X(t-\Delta t)-2X(t)+X(t+\Delta t)) \qquad (1)$$

where X represents a model field.
This requires knowledge of the unfiltered fields at three time levels. Since the rest of the model needs only two time levels, it would be an extravagent use of memory to store a third level simply for time filtering purposes. Fortunately, the recursive filter

$$F : X(t) \rightarrow X(t) + (\nu/2)(F(X(t-\Delta t))-2X(t)+X(t+\Delta t)) \qquad (2)$$

requires only two time levels and has similar phase and amplitude characteristics to the non-recursive filter for small values of ν (Asselin 1972).

To illustrate the implementation of the filter, consider a single grid point and suppose that the FORTRAN variable A contains the value of the quantity X at time t and that B contains the value of X at time $t-\Delta t$. The integration is assumed to have proceeded for more than one time step so that B has already been filtered. In the notation of (2), A contains X(t) and B contains $F(X(t-\Delta t))$. Suppose that a third variable H contains the increment which will step B forward to time $t+\Delta t$. The following sequence performs a time step and simultaneously applies the time filter.

```
A = (1-ν)A  +.5*ν*B
B = B + H
A = A +.5*ν*B
```

The result of these operations is that B contains $X(t+\Delta t)$ and A contains $F(X(t))$. A and B are then swapped in the usual manner before the next time step is performed.

In the model, increments are collected for all grid points at a single latitude of the grid before being added to the update fields, so that A, B and H above correspond to two dimensional arrays.

On the Use of an Increment Array

The stratosphere-mesosphere model collects increments from all its time stepping routines in an array whose contents are added to the "update" fields only when all the increments for a given latitude have been calculated.

Since the model time-steps latitudes independently, the array of increments is two dimensional and does not significantly increase the size of the model. Furthermore, the use of an increment array has three important advantages over the alternative of incrementing the "update" fields directly from each of the time-stepping routines.

Firstly, by using an increment array, the model can apply a time filter as described in the previous section of this note. This reduces the amplitude of undesirable computational and high frequency gravity wave modes. Secondly, the increments may be filtered or fourier truncated before being added to the model's fields. As a method of controlling linear stability near the poles, this is preferable to filtering the fields of the model, although still not ideal (Takas and Balgovind 1983).

The third advantage of using an increment array is increased accuracy of physical parametrizations. To demonstrate this suppose we have two physical parametrizations in the model which calculate the rate of change of the model's fields. We may represent these parametrizations by functionals A and B. Ignoring all other processes, the rate of change of the model's fields, represented by X, is given by

$$\frac{\partial X}{\partial t} = A(X) + B(X)$$

Suppose further that A and B require the use of a forward time step for stability. In the model, this means that increments are calculated from the "update" fields rather than the "current" fields. When an increment array is used, the integration is performed as follows

$$X(t+\Delta t) = X(t) + \Delta tA(X(t)) + \Delta tB(X(t))$$

However, when increments are added to the update fields as soon as they are calculated, the integration gives (assuming A is applied first)

$$X(t+\Delta t) = X(t) + \Delta tA(X(t)) + \Delta tB(X(t)+\Delta tA(X(t)))$$

This can lead to inaccurate integration for three reasons: if A is e.g. a solar heating scheme, then the input to B may be consistently biassed, leading to systematic errors in the calculations for process B; if B is highly non-linear then the small errors in its input produced by adding the increments from A may lead to large errors in the increments calculated by B; lastly, even if B is linear so that we may write the above as

$$X(t+\Delta t) = X(t) + \Delta tA(X(t)) + \Delta tB(X(t)) + \Delta t^2B(A(X(t))),$$

the formally second order error may be large compared with the combined effect of A and B if they represent processes which tend to balance, e.g. solar heating and long-wave cooling.

Restarting and Data Spooling

The stratosphere-mesosphere model is a large program which requires large amounts of computer time and generates large amounts of data. Unfortunately, it is not sufficient simply to suspend the execution of such a program to allow time-critical tasks such as operational weather forecasting to be performed, since suspending the program does not free the large amount of disk storage space typically required whilst the program is running. Instead, some means must be provided whereby a computer operator can stop the model in a controlled fashion which releases resources and allows the model to be restarted, preferably automatically, at a later time. The system adopted for the stratosphere-mesosphere model is as follows.

Immediately before writing each dump to the output file, the model tests whether it has been asked to stop by a computer operator. (The method of communication between operator and model varies between operating systems, but the facility is widely available.) If the model has not been asked to stop, the dump is written to the output file and the integration continues. Otherwise, the dump is written to a permanent file called the "restart file" and the model stops. The output file is then copied to magnetic tape and deleted from disk and a job is automatically submitted to the job queue. When it runs, the job creates a new, empty output file and continues the integration, using the contents of the restart file as initial data.

Note that it is not possible to eliminate the output file by writing directly to magnetic tape as this would require a tape drive to be allocated to the model throughout the integration.

The output file is limited in size by the amount of disk space available and the time required to transfer its contents to tape. Were the model allowed to integrate indefinitely, it would eventually fill the output file and the integration would fail. Consequently, the model keeps track of the amount of data which has been written to the output file and uses this information to calculate the space remaining in the output file before it writes a dump. If there is insufficient space for another dump, the model behaves as if the operator had asked it to stop. This has the effect of copying the dumps to tape and restarting the integration with an empty output file.

The method outlined above has the disadvantage that the model continues integrating after it has been asked to stop until it writes a dump. At high resolution or on some computer installations, this will result in an unacceptably slow response to the operator's request. However, if the model is stopped between dumps then both time levels must be saved to allow the model to restart with a leapfrog time-step. Otherwise, by asking the model to stop, the operator would force the model to perform a forward time-step at unpredictable times during the integration. This would make it difficult to repeat experiments or to compare perturbation experiments with their controls.

Whilst it is possible to explicitly save both time levels in the restart dump, a better solution to the problem of restarting between dumps is to use the checkpoint/restart facility which is available on several machines. A checkpoint is a call to a subroutine which produces a file containing a copy of the entire program, including the values of all variables and arrays. Subsequently, the program may be restarted from this checkpoint file. Execution continues from the statement following the call

to the checkpoint subroutine and, since the values of all variables have been saved, the process of restarting cannot affect the results of the integration. This is an important advantage over explicit restarting.

The checkpoint/restart technique is the standard method used for models run on the ECMWF Cray-XMP computer. It is recommended that future versions of the stratosphere-mesosphere model be modified to use the technique.

Description of a Model Run

This section describes the tasks performed by the main subroutines of the model during the course of an integration.

Fig. 1 illustrates the interconnection of modules in the stratosphere-mesosphere model. It is recommended that the reader consult this diagram when reading the following.

A model run begins in MODEL, the main program. This routine opens the files required by the model and then calls subroutines which read, print and test the values of the 512 constants stored in the IPARMS and RPARMS arrays, and the card images which determine the dates of the beginning and end of the integration, etc.

MODEL then calls STRTUP which reads the initial data by calling RDINIT, reads "future" boundary data by calling UPDBDY, and creates a header for output dumps by calling MODHED.

Next STRTUP calls TMLOOP which controls the integration process. A loop in TMLOOP integrates forward in time by repeated calls to TIMSTP. The lower boundary height field is stepped forward in time by calls to UPDBDY. At each iteration of the loop, the function DMPTIM tests whether a dump should be written to the output file. Dumps are written by calls to WRDUMP.

Before writing an output dump, a call to ASKOP tests whether the computer operator has asked the model to stop. If so, subroutine OPSTOP is called which writes a dump to the restart file before stopping.

TMLOOP also tests for the end of the integration as determined by the end date supplied in file CARDS. When the end date is reached, subroutine ENDRUN is called which writes the final dump to the output file and stops the model.

As well as stopping on request or on reaching the end date, the model will also stop if the output file becomes full. WRDUMP tests for this condition using the function FULFIL and calls RESTRT if the file is full. RESTRT writes the dump to the restart file and stops the model.

Individual time-steps are performed by TIMSTP. This routine controls the model's loop over latitude. Subroutine UPDLAT is called once for each latitude of the model grid to time-step the fields at that latitude. Calls to PAGOUT and PGNEXT perform implicit paging if the model is larger than the computer's real memory. UPDLAT also contains the call to FILTNS which applies the meridional filter to the model fields. FILTNS is called outside the loop over latitude and can therefore not be used in a paging model. It is hoped that future versions of the model will replace FILTNS with a diffusion scheme called from within UPDLAT.

UPDLAT time-steps a single latitude of the model by calling CHOP, FILTEW, DNAMIC, NEWTON, RAYLEY and UPDTMF, which Fourier truncate, zonally filter, calculate increments due to dynamics, radiation and friction, time filter and add increments to the model fields.

This section has only described the more important subroutines of the model. However, the purpose of each routine in the model is described by comments at the start of the routine and also (briefly) in Appendix I of this note. It is hoped that the reader will have little difficulty in understanding the routines not explicitly documented here.

The Main Arrays, Variables and Parameters

Whilst there are hundreds of variables, parameters and arrays declared in the model, a few are particularly important and occur in a large proportion of the subroutines. For clarity, these have been given the same names in each of the subroutines. They are listed and described below.

Parameters

MERIDS LATS LEVELS LPGPAR LENLAT

A PARAMETER statement specifying these parameters occurs in most of the model's subroutines. MERIDS, LATS and LEVELS determine the resolution of the grid, MERIDS being the number of grid points in the E-W direction (not including any "wrap around" points), LATS being the number of grid points in the N-S direction and LEVELS being the number of levels for which temperature and horizontal wind components are stored.

The paging scheme requires the three dimensional data to be divided into units which are a multiple of the computer's page size. There is one such unit for each latitude of the grid. The parameter LENLAT determines how many array elements are allocated to each latitude of the grid and is a multiple of LPGPAR, the number of elements in one page.

If the paging scheme is not used, LPGPAR may be set to one to avoid wasting space although, on a virtual storage machine, it is probably still worth setting LPGPAR to the page length.

Note that in the base code, LENLAT is calculated by rounding up the space required for the temperature and horizontal wind components to a multiple of LPGPAR. In some versions of the model, however, LENLAT is longer in order to accomodate three dimensional ozone fields, Curtis matrices, etc.

Arrays

DOMAIN WORK HINCR HEADER

IPARMS RPARMS ZNOW

The three dimensional fields are stored in "DOMAIN". In the base code model, these are temperature and the horizontal wind components at the two time levels required for leapfrog integration.

The storage of data in "DOMAIN" is best illustrated by the following FORTRAN example:

```
                    .
                    .
                    .
      DIMENSION DOMAIN(LENLAT,LATS)
                    .
                    .
                    .
      CALL XXXX (DOMAIN(1,J),ITIME)
                    .
                    .
                    .
      END
      SUBROUTINE XXXX (TUV, ITIME)
                    .
                    .
                    .
      DIMENSION TUV(MERIDS+2,LEVELS,3,2)
                    .
                    .
                    .
      T = TUV(I,K,1,ITIME)
      U = TUV(I,K,2,ITIME)
      V = TUV(I,K,3,ITIME)
                    .
                    .
                    .
      END
```

The array "TUV" in subroutine "XXXX" in the above example contains the values of temperature and horizontal wind components at latitude $\pi(1-(2*J-1)/LATS)/2$ radians north. The three assignment statements set T, U and V to the values of temperature, eastwards wind and northwards wind respectively at the point on the Kth level above the lower boundary and at $2\pi(I-1)/MERIDS$ radians east. Notice that each row of values contains two

"wrap around" points i.e. the first two values in each row are repeated at the end of the row. This greatly simplifies finite differencing and filtering in the zonal direction.

The variable "ITIME" determines which of the two time levels is referenced, and is described below.

It is possible for other three dimensional fields to be stored in "DOMAIN". Values of these fields for each latitude of the grid are always stored after the temperature and wind fields so that the above example is valid for all versions of the model.

The "WORK" array, as its name implies, is used to store intermediate results. It is large enough to store the values at one latitude of twelve fields. However, subroutine "UPDLAT" uses the last quarter of the array to accumulate increments from the routines it calls. This area of "WORK" is passed separately and called "HINCR" in these routines and "WORK" is given correspondingly smaller dimensions.

The name "HEADER" is used for any array which is used to store a header record from initial, boundary or output dumps. They are the only arrays in which different types of data are mixed and contain real, integer and character values.

One of the shortcomings of the FORTRAN language is that it does not allow data of various types to be combined into a single entity except as a common block (which cannot be passed via an argument list). It is therefore necessary to resort to trickery to smuggle data of one type into an array of a different type or to use separate arrays to store the various types. The latter approach is preferable. However, in the

stratosphere-mesosphere model this approach was not possible without changing the "BUFIN" and "BUFOUT" routines. As these routines are used by several other programs (diagnostics, initialisation, etc), this was not deemed practical. Hopefully, future FORTRAN standards will rectify the problem and bring the language into line with most other commonly used programming languages by providing user-defined compound types. Until then, "HEADER" arrays are one of the few areas of the model which require careful attention when adapting the model for a different word length or machine. To facilitate such adaptation, "HEADER" arrays are always explicitly typed as REAL since on most machines REAL variables and INTEGER variables require the same amount of storage space. The base code model assumes that four bytes are used for both types.

Arrays "IPARMS" and "RPARMS" hold values of physical constants and constants which control the integration such as the pressures of the boundary levels and the length of the time step. Each array holds 256 values. Real values are stored in "RPARMS" and integer values are stored in "IPARMS". Note that there are no global arrays of e.g. cosine of latitude or pressures at every level. Where such arrays are required, they are calculated from the constants in "IPARMS" and "RPARMS" on first entry to the routine which needs them by a call to a subsidiary routine. In this way, the number of arguments passed between routines is kept managebly small.

Throughout an integration, the model requires a field of geopotential height at the lower boundary. This is stored in "ZNOW". The following FORTRAN example demonstrates the storage of values in "ZNOW":

```
      .
      .
      .
DIMENSION ZNOW(MERIDS+2,LATS)
      .
      .
      .
H = ZNOW(I,J)
      .
      .
      .
END
```

In the above, the variable H is assigned the value of geopotential height at 2π(I-1)/MERIDS radians east and π(1-(2J-1)/LATS)/2 radians north. Notice that each row contains two "wrap around" points which repeat the first two values of the row.

Variables

ISEC	IDAY	IMON	IYEAR
ITIME	PBASE	PTOP	
ISTSEC	ISTDAY	ISTMON	ISTYR
INDSEC	INDDAY	INDMON	INDYR
IDMSEC	IDMDAY		

The stratosphere-mesosphere model represents time using a real calendar as seconds past midnight on a day of a month of a year. This requires four integers. The time stepping routines use "ISEC", "IDAY", "IMON" and "IYEAR" to store the time and date of the fields at the "current" time level (i.e. the fields which have most recently been updated by the leapfrog integration). The date of the initial data is stored in "ISTSEC", "ISTDAY", "ISTMON" and "ISTYR" and the date at which the model is

required to stop is stored in "INDSEC", "INDDAY", "INDMON" and "INDYR". The interval between output dumps is represented in seconds and days in "IDMSEC" and "IDMDAY". In all cases the year is represented as years A.D.

The leapfrog integration uses data at two time levels which may be called the "update" and "current" time levels. To perform a timestep, increments are calculated from the "current" fields and added to the "update" fields. Before the next timestep, the fields are swapped so that the "current" fields for one timestep are the "update" fields for the next timestep.

For each latitude of the grid, the "current" and "update" fields are stored contiguously. To avoid physically swapping them before every timestep, a pointer "ITIME" is used. "ITIME" takes the value 1 or 2 at alternate timesteps. A value of 1 indicates that the "current" fields precede the "update" fields.

Finally, the variables PBASE and PTOP occur frequently in the code and are always used to store the pressures of the model's lower boundary and top level.

The Dynamics Scheme

The dynamics scheme is the most computationally expensive section of the model. As such, it has been necessary to sacrifice a degree of clarity and modularity to achieve efficient code. Nevertheless, it has proved possible to divide the calculations into eight subroutines, each of which corresponds to an identifiable group of terms of the finite difference equations. These are respectively: zonal advection of temperature and wind, zonal height gradient, calculation of vertical velocity, adiabatic heating, vertical advection, meridional height gradient, meridional advection, and the coriolis terms.

Because the finite difference equations are expressed in "flux form", it would be possible to save values of meridional advective fluxes calculated at the southern boundary of a grid box for use at the northern boundary of a neighbouring grid box. This method has not been used in the stratosphere-mesosphere model, partly because of the extra complexity introduced by the fourth order differencing, but also because the method requires the latitude rows of the model's grid to be processed sequentially, thereby making the model difficult to adapt for use on a multiprocessing computer.

Instead, the model makes use of the fact that most of the calculations required to compute the advective fluxes are also required to compute vertical velocity and meridional height gradients. The overhead incurred by not saving the fluxes between latitudes is therefore not large.

Two general features of the scheme will be noted here before discussing separately each of the subroutines which make up the scheme. Firstly, several of the routines perform meridional differencing which requires knowledge of values of temperature, wind or lower boundary height

from across the poles. This complicates array indexing and requires the use of cross-polar "shuffling" routines CRSPOL and CPOLE. In each case, however, it has been possible to separate the shuffling and array indexing from the calculations, which are performed in subsidiary routines. Secondly, several of the routines compute intermediate reuslts which are useful in other routines of the dynamics scheme. These are passed between routines via the WORK array. Such use of side effects complicates the scheme and would be bad programming practice were it not for the overriding requirement for speedy code in this section of the model.

DNAMIC

DNAMIC is the top level routine of the dynamics scheme. It calls each of the eight routines which calculate temperature and wind increments. On the first call only it calls SETDYN which calculates arrays DCOEFF and DCONST of coefficients for use in the rest of the scheme. Where appropriate, DNAMIC performs the array indexing which allows routines such as DYNEWA to be oblivious of the value of ITIME or the latitude of the grid for which calculations are performed.

SETDYN

This routine calculates coefficients for the dynamics scheme from the constants in arrays IPARMS and RPARMS. Constant coefficients are stored in array DCONST and coefficients which vary with latitude are stored in DCOEFF. None of the coefficients varies with longitude or pressure.

DYNEWA

Zonal advection is calculated using the relation

$$\delta_\lambda(\overline{u}^{\lambda}\overline{X}^{\lambda}) = \delta_\lambda(\overline{uX}^{\lambda}) + u\,\delta_\lambda\overline{X}^{\lambda} + X\,\delta_\lambda\overline{u}^{\lambda}$$

where X represents temperature or a wind component. This has two advantages. Firstly, the differences required for the second term RHS may be saved for re-use in other calculations. Secondly, the last term of the above need not be calculated since it cancels with similar terms from the meridional and vertical advection equations via the continuity equation. The fourth order differences are calculated using the factorisation technique (Fisher 1986a) whereby the operator (1,-8,0,8,1) is represented by (1,0,-1), (-4+√15,1) and (-4-√15,1) applied in succession.

DYNEWZ

DYNEWZ integrates the hydrostatic equation to calculate the zonal height gradient at every level of the model. The factorisation technique is used to calculate the gradient at the lower bounday and the differences of temperature which were calculated in DYNEWA are used to integrate upwards. By integrating the gradient of height rather than height itself, the model removes the need for a three dimensional array of height, ensures that the calculations at adjacent latitudes may be performed concurrently, and reduces rounding errors.

DYNNSW

This routine calculates "vertical velocity", $\dot{s}$. As with all the routines which perform meridional differencing, the array indexing and shuffling of cross-polar data are separated from the main calculations. Vertical velocity is therefore calculated in the subsidiary routine CLCNSW. Meridional differences of temperature and wind components are calculated and stored in WORK. The meridional differences of meridional wind are then combined with the zonal differences of zonal wind produced by DYNEWA and then integrated down from the top of the model.

DYNADI

Adiabatic heating is calculated from the model's temperature field and the "vertical velocity" produced by DYNNSW.

DYNVAD

The calculation of vertical advection is divided into two terms:

$$\frac{1}{p}\,\delta_s(p\dot{s}\bar{X}^s) = (X_{k+1}(p\dot{s})_{k+1/2} - X_{k-1}(p\dot{s})_{k-1/2})/2p_k\Delta s + \frac{1}{p}\,X_k\delta_s(p\dot{s})$$

where X represents temperature or a horizontal wind component. The last term RHS is not calculated as it cancels with similar terms in the zonal and meridional equations. The values of temperature and wind at the lower boundary are calculated in DYNVAD by linear extrapolation from the lowest two levels of the model. Note that, although pressure appears in the above expression, it occurs only as a ratio of pressures at adjacent whole and half levels. This ratio is a constant for the model.

DYNVAD is quite long-winded. This is because, at this stage of the calculations, WORK holds several intermediate calculations and there are not three contiguous areas of work space which would allow a simple loop over temperature and wind. Consequently, each section of code is repeated three times.

DYNNSZ

Being a meridional differencing routine, DYNNSZ merely performs array indexing and shuffling of data. The calculations are performed separately in CLCNSZ. Meridional height gradients are integrated from the lower boundary in a similar manner to the zonal height gradients calculated by DYNEWZ. Because of the special form of the expression for the gradient, however, the meridional temperature gradients must be modified by the addition of a weighted central value before being integrated.

DYNNSA

DYNNSA differs from the other meridional differencing routines in that it performs some of the calculations in addition to array indexing and shuffling data. The meridional advection is calculated using the relation

$$\delta_\phi(\bar{u}^\phi\bar{X}^\phi\cos\phi = \delta_\phi(\overline{uX}^\phi)\cos\phi + u\delta_\phi\bar{X}^\phi\cos\phi + X\delta_\phi\bar{u}^\phi\cos\phi$$

The second term RHS is calculated in DYNNSA, thereby freeing an area of WORK for data shuffling. The third term is not calculated as it cancels with similar terms in the zonal and vertical equations. The first term is calculated in CLCNSA.

DYNCOR

DYNCOR calculates the wind increments due to the coriolis and curvature effects.

Shapiro Filters

The model controls non-linear instability by applying eighth order Shapiro filters (Shapiro 1970) in the zonal and meridional directions. The filters are implemented using the factorization technique discussed in a separate note (Fisher 1986a).

Zonal Filtering

The zonal filter routine FILTEW illustrates the advantages both of the factorization technique and the use of two "wrap around" points on the end of each row. The entire process of applying the 17-point filter to the model's temperature and wind fields at one latitude of the grid is performed by a mere nine executable statements. The routine applies each of eight 3-point filter operators in turn with two vector loops and restores the values at the first point and second wrap-around point of each row using subroutine RESEND. Finally the fields are multiplied by 1/65536, suitably adjusted to give exactly unity response at infinite wave length.

Meridional Filtering

Meridional filtering is more complicated than zonal filtering because of the more complex boundary conditions at the poles. The eight 3-point operators used in FILTEW must be combined in pairs to give four 5-point symmetric operators. These are applied in alternate north-to-south and south-to-north passes by subroutines FLTNTS and FLTSTN which are called by the main meridional filtering routine FILTNS. North-to-south passes are performed by FLTNTS. Cross-polar values of temperature and wind are calculated from values at the first two latitudes of the grid and stored in WORK. The filter operator is then applied to the grid, filtered values being stored in the area of DOMAIN (or WORK) which held values two rows to the north of (or over the pole from) the centre of the operator. The

south-to-north passes are performed by FLTSTN in a similar manner to the north-to-south passes. In FLTSTN, however, filtered values are stored two rows to the south of the centre of the filter operator. The net meridional shift after an equal number of calls to FLTNTS and FLTSTN is therefore zero.

Data Requirements

Input to the model consists of five files containing initial data, boundary data, parameters, dates, and "tidy-up" JCL, respectively. Of these, the initial and boundary data files are read using the BUFIN routine described elsewhere in this note. It is assumed that the reader is familiar with this routine.

File INITIAL (Unit 10)

Initial data is read by the BUFIN routine. The header read by this routine must contain the date for which the data was valid in the 517th to 520th elements as four integers giving the number of seconds past midnight, the day, month and year. Apart from these elements, the second half of the header should contain 256 integers followed by 256 real numbers corresponding to the model's IPARMS and RPARMS arrays. These are checked for compatability with the model by subroutine TSTINI. It is conventional to store a title for the program which created the data in the first 80 bytes of the header and a title for the data in the second 80 bytes.

The initial data dump must contain the following fields.

BDY	This field is read into the model's ZNOW array. See elsewhere in this note for the format of data in this array.
TBAR	Zonal mean radiative equilibrium temperatures (K) in north-to-south, bottom-to-top order.

TUVnnn (Where nnn represents a number, padded with leading blanks.)	One of these fields is required for each latitude of the model. Each contains temperature and wind components for latitude 'nnn' of the grid, latitudes being numbered in north-to-south order. The format of data within each field is as described elsewhere in this note for the corresponding sub-arrays of DOMAIN.

The initial data file may contain more than one dump. These are assumed to be in chronological order. The model skips past any dump with a validity date which is earlier than the required start of the integration. The first dump in the file which is valid on or later than this date is used as initial data.

File BOUNDARY (Unit 12)

Boundary data is similar in format to initial data. Only the BDY field is required for a boundary data dump, but the boundary file will usually contain several dumps. The header of each dump is tested for compatibility with the model by TSTBDY. The boundary dumps must be in chronological order of validity date and at least one dump must be valid at a later date than the end of the integration. The model skips past any dumps which are valid on or before the date of the initial data.

File CARDS (Unit 13)

File CARDS contains four records which are read using FORTRAN list-directed input. The first record consists of an experiment number followed by a title for the experiment of up to 72 characters enclosed in

single quotes. These are inserted into the header of each dump output from the model. The second and third records specify the dates of the start and end of the integration, each as four integers giving the number of seconds past midnight, day, month and year. The fourth record specifies the interval between output dumps in seconds and days.

File TIDYJCL (Unit 14)

Whenever the model stops integrating, a subsequent step in the job which is running the model submits the contents of file RSTTJCL to the job queue. Normally, this file contains the JCL which restarts the integration. However, when the integration has been completed, the model must not be restarted. RSTTJCL is therefore overwritten with the contents of TIDYJCL at the end of an integration by subroutine CPYJCL. TIDYJCL may contain JCL for a job which deletes unwanted files, produces diagnostics, etc.

File PARAMS (Unit 16)

File PARAMS contains 256 integers and 256 real numbers which are read from card images using (8I10) and (8G10.3) formats into arrays IPARMS and RPARMS. Appendix VI lists the constants to which the elements of these arrays must be set.

Output from the Model

The model writes data to four files: printed output, output dumps, restart dump and restart JCL.

Printed Output

All printed output is written using the FORTRAN PRINT statement to the default unit for such output. The amount of output which is produced is small.

File OUTPUT (Unit 11)

Output dumps are written to file OUTPUT using the BUFOUT routine. The header for each dump is produced by substituting the data title and the last 512 words of the header from the initial data with the experiment title, IPARMS and RPARMS. The date and experiment number are also inserted in the appropriate places in the header. The fields written to each dump are as described for the initial data. Indeed, output dumps may be used as initial data or boundary data for other integrations.

Cyber 205 users should not use the name OUTPUT for the output file as this is reserved for the printed output file.

File RSTTJCL (Unit 17)

This file normally contains the JCL for a restarted model run. The model writes to this file at the end of an integration to prevent the model from restarting.

File RSTTDMP (Unit 15)

When the model stops before the end of an integration because the output file is full or the computer operator has asked it to stop, the model writes a dump to RSTTDMP. The dump is in the same form as an output dump and is used as initial data when the model restarts.

Conservation Properties

Mass

Conservation of mass is trivially assured as a consequence of the model's isobaric co-ordinate system. The mass of a grid box (i.e. the volume surrounding a grid point whose boundaries are equidistant in longitude, latitude and log (p/p_o) from neighbouring points) is constant and given by

$$m = pa^2\cos\phi\ \Delta\lambda\Delta\phi\Delta s/g \qquad (1)$$

A Useful Result

Before proceeding to derive the energy equations for the model, it is useful to note the following result, the proof of which involves only simple algebraic manipulation and is not included here.

Lemma

Let $X = X(x,\ldots)$, $Y = Y(x,\ldots)$, $Z = Z(x,\ldots)$

Then $$\delta_x(\widetilde{XY}^x Z) = X\delta_x(\bar{Y}^x Z) + Y\delta_x(\bar{X}^x Z) - XY\delta_x Z \qquad (2)$$

A special case of (2) is obtained by setting $Z = 1$ and $Y = X$, giving

$$X\delta_x\bar{X}^x = \delta_x(\widetilde{XX}^x/2) \qquad (3)$$

Kinetic Energy

Following the standard derivation of the continuous kinetic energy equation, we consider:

$$u\delta_t\bar{u}^t + v\delta_t\bar{v}^t \qquad (4)$$

Using the lemma (3), this may be written as

$$\delta_t[(\widetilde{uu}^t + \widetilde{vv}^t)/2]$$

We therefore define the finite difference analogue of kinetic energy per unit mass as

$$K = (\widetilde{uu}^t + \widetilde{vv}^t)/2$$

Note that there is no contribution to K from the vertical velocity. This is a result of using the hydrostatic approximation.

Analysis of the fourth order equations of the model is simplified by observing that they may be expressed as a linear combination of second order accurate equations solved on two different grids. It is evident from (4) that the kinetic energy equation for the model may also be regarded as a linear combination of the corresponding second order equations. Therefore, the following derivation will only consider the second order equations which are as follows.

$$\delta_t \bar{u}^t + D(u) + (f + \frac{u}{a}\tan\phi)v + (a\cos\phi)^{-1}\delta_\lambda \bar{\Phi}^\lambda = 0 \tag{5}$$

$$\delta_t \bar{v}^t + D(v) + (f + \frac{u}{a}\tan\phi)u + (a\cos\phi)^{-1}[\delta_\phi(\bar{\Phi}^\phi \cos\phi) - \Phi\delta_\phi\cos\phi] = 0 \tag{6}$$

$$\delta_t \bar{T}^t + D(T) - \frac{\kappa(\overline{p\dot{s}}^s)T}{p} = 0 \tag{7}$$

$$D(1) = 0 \tag{8}$$

$$\delta_s \Phi + R\bar{T}^s = 0 \tag{9}$$

where $s = \log(p/p_o)$

and

$$D(X) = (a\cos\phi)^{-1}[\delta_\lambda(\bar{u}^\lambda \bar{X}^\lambda) + \delta_\phi(\bar{v}^\phi \bar{X}^\phi \cos\phi)] + \frac{1}{p}\delta_s(p\dot{s}\bar{X}^s)$$

Combining (5) and (6) and subtracting $^1/_2(u^2 + v^2)\,D(1)$ (which is zero) we have

$$\delta_t K + A + B = 0 \tag{10}$$

where

$$A = uD(u) - {}^1/_2u^2D(1) + vD(v) - {}^1/_2v^2D(1) \tag{11}$$

and $B = (a\cos\phi)^{-1}[u\delta_\lambda\bar{\Phi}^\lambda + v\delta_\phi(\bar{\Phi}^\phi\cos\phi) - v\Phi\delta_\phi\cos\phi]$ (12)

Equation (11) may be simplified by applying the lemma (2) to give

$$A = (a\cos\phi)^{-1}[\delta_\lambda((\widetilde{u}\widetilde{u}^\lambda + \widetilde{v}\widetilde{v}^\lambda)\bar{u}^\lambda/2) + \delta_\phi((\widetilde{u}\widetilde{u}^\phi + \widetilde{v}\widetilde{v}^\phi)\bar{v}^\phi\cos\phi)/2]$$

$$+ \frac{1}{p}\ \delta_s((\widetilde{u}\widetilde{u}^s + \widetilde{v}\widetilde{v}^s)\dot{p}s/2) \qquad (13)$$

Thus A corresponds to the rate of change of kinetic energy as a result of fluxes through the boundaries of the grid box. Unfortunately, each of the fluxes is represented by a different approximation to the kinetic energy resulting in spurious generation of kinetic energy in the grid box. However, on multiplying (13) by the mass of a grid box (1) and summing over all grid points, we find that the contribution to the global kinetic energy from A reduces to a flux through the lower boundary.

$$\text{ie. } \sum_{\lambda,\phi,s} mA = \sum_{\lambda,\phi}[(\widetilde{u}\widetilde{u}^s + \widetilde{v}\widetilde{v}^s)\dot{p}s/2]_{1/2}(m/p)_1 \qquad (14)$$

where the subscript 1 indicates the lowest level of the model and $^1/_2$ indicates half a level below the lowest level.

Equation (12) represents the work done by the pressure gradient force, being a finite different approximation to $\underline{v}.\nabla\Phi$ (Again, there is no contribution from the vertical velocity because of the hydrostatic approximation). Using the lemma (2), we may rewrite (12) as

$$B = (a\cos\phi)^{-1}[\delta_\lambda(\widetilde{u\Phi}^\lambda) + \delta_\phi(\widetilde{v\Phi}^\phi\cos\phi)] \qquad (15)$$

$$- (a\cos\phi)^{-1}\Phi[\delta_\lambda\bar{u}^\lambda + \delta_\phi(\bar{v}^\phi\cos\phi)]$$

ie. $$B = (a\cos\phi)^{-1}[\delta_\lambda(\widetilde{u}\Phi^\lambda) + \delta_\phi(\widetilde{v}\Phi^\phi\cos\phi)] + \frac{\Phi}{p}\delta_s(p\dot{s}) \qquad (16)$$

where the continuity equation (8) has been used to eliminate the horizontal divergence. The first term in (16) is a flux of geopotential into the grid box. Note that, on multiplying by the mass of a grid box (1) and summing over horizontal grid points, the flux term vanishes.

The second term of (16) is best treated by multiplying by the mass of a grid box (1) and summing vertically, giving

$$\sum \Phi\delta_s p\dot{s}\, a^2\cos\phi\ \Delta\lambda\Delta\phi\Delta s/g \qquad (17)$$

After a little manipulation, and with the aid of the upper boundary condition which sets $\dot{s} = 0$ half a level above the top of the model, we may write (17) as

$$[\sum p\dot{s}\delta_s\Phi + \Phi_o(p\dot{s})_{1/2}]\, a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g \qquad (18)$$

Here, the summation is over all half levels between the lower boundary and the top of the model. The subcripts 0 and $^1/_2$ indicate the lower boundary and half a level above it respectively.

Finally, replacing the vertical difference of geopotential using the hydrostatic equation (9), we have

$$[\sum_g Rp\dot{s}\bar{T}^s + \Phi_o(p\dot{s})_{1/2}]a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g \qquad (19)$$

Hence

$$\sum_g mB = \sum[\delta_\lambda(\widetilde{u}\Phi^\lambda) + \delta_\phi((\widetilde{v}\Phi^\phi)\cos\phi)]a^2\Delta\lambda\Delta\phi\Delta s/g$$
$$+ \sum_g Rp\dot{s}\bar{T}^s\, a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g$$
$$+ \Phi_o(p\dot{s})_{1/2}\, a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g \qquad (20)$$

Enthalpy

Enthalpy per unit mass is defined as $H=C_pT$. The model's thermodynamic equation is therefore also the enthalpy equation for the model. Multiplying (7) by C_p, we have

$$\delta_t H + D(H) - \underline{R\overline{(p\dot{s})}^s}\underline{T} = 0 \qquad (21)$$

As was the case for the kinetic energy equation, the enthalpy equation consists of two terms, representing advection and generation of energy. A mass weighted sum over grid points of the advection term leaves only a flux through the lower boundary.

$$\sum_{\lambda,\phi,s} D(H) = \sum_{\lambda,\phi} \bar{H}_1^s (p\dot{s})_{1/2}(m/p)_1 \qquad (22)$$

where the subscripts 1 and $^1/_2$ are as described for equation (14).

The generation of enthalpy is represented by the last term in equation (21). To examine this further, we multiply by the mass of a grid box and sum vertically to give

$$\sum_s - R(\overline{p\dot{s}}^s)T\, a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g \qquad (23)$$

which, after a little manipulation may be written as

$$[\sum_s - R(p\dot{s})\bar{T}^s + R(p\dot{s})_{1/2}T_o]a^2\cos\phi\Delta\lambda\Delta\phi\Delta s/g \qquad (24)$$

Note that the vertical sum in (24) is identical apart from sign to a term in equation (20) and therefore may be regarded as a conversion between kinetic energy and enthalpy. The terms cancel and therefore make no contribution to the rate of change kinetic energy plus enthalpy.

The model is therefore energy conserving in the usual sense of conserving the sum of kinetic energy and enthalpy. Strictly, however, the model does not conserve energy since it is not possible to equate enthalpy with the sum of potential and internal energies. Fisher (1987) shows how this shortcoming may be overcome by making minor changes to the finite difference equations.

References

Asselin, R. 1972 Frequency Filter for Time Integrations. Monthly Weather Review Vol. 100 No. 6 pp 487-490.

Barker, J.W. 1985 Gravity Wave Parametrization in the Meteorological Office Stratosphere-Mesosphere Model. (Unpublished).

Butchart, N., Clough, S.A., Palmer, T.N. and Trevelyan, P.J. 1982 Simulations of an Observed Stratospheric Warming with Quasi-geostrophic Refractive Index as a Model Diagnostic. Q.J.R.M.S. Vol. 108 No 457 pp 475-502.

Deque, M., Cariolle, D. 1986 Some Destabilizing Properties of the Asselin Time Filter. Monthly Weather Review Vol. 114 No. 5 pp 880-884.

Fisher, M. 1986a Efficient Implementation of Multipoint Operators. Met O 20 Internal Note No. 59.

Fisher, M. 1986b A Portable UPDATE-like Source code Pre-processor. Met O 20 Internal Note No. 58.

Fisher, M. 1987 Energy Conservation in the Met O 20 Climate and Stratosphere-Mesosphere Models. (To be issued as a DCTN)

Foreman, A.M. 1984 Cyber 205 Data Conversion Routines, Met O 12 Program Documentation.

Hinds, M.K. 1982 Forward Time-Steps and Computational Instability. Met O 20 Internal Note No. 5.

Holton, J.R. 1972a An Introduction to Dynamic Meteorology. International Geophysics Series, Vol. 16 Academic Press.

Holton, J.R. 1972b A Semi-Spectral Numerical Model for Wave-Mean Flow Interactions in the Stratosphere: Applications to Sudden Stratospheric Warmings. J. Atmospheric Science Vol. 33. No. 8 pp 1639-1649.

Kalney-Rivas, E. 1978 High-Latitude Truncation Errors of Box-Type Primitive Equation Models. Monthly Weather Review Vol. 104. No. 8 pp 1066-1069.

Maynard, J. 1972 Modular programming. Pub. Butterworths ISBN 0 408 70287 7.

Murgatroyd, R.J. 1971 Dynamical Modelling of the Stratosphere and Mesosphere. In "Mesospheric Models and Related Experiments". Proc 4th ERSIN-ESLAB Symposium Frascatti, Italy 6-10 July 1970 pp 104-121

Shapiro, R. 1970 Smoothing, Filtering, and Boundary Effects. Reviews of Space Physics Vol. 8 No. 2 pp 359-387.

Shine, K.P.	1984	Solar Radiative Heating Calculations for the Meteorological Office Stratosphere-Mesosphere Model. (Unpublished).
Shine, K.P.	1985a	Ozone Climatology for the Meteorological Office Stratosphere-Mesosphere Model. (Unpublished).
Shine, K.P.	1985b	An Infrared Radiation Scheme for the Meteorological Office Stratosphere-Mesosphere Model. (Unpublished).
Takas, L.L., Balgovind, R.C.	1983	High Latitude Filtering in Global Grid Point Models Monthly Weather Review. Vol. 111 No. 10 pp 2005-2015.
Temperton, C.	1976	An All-purpose Mixed-Radix Fast Fourier Transform Program. Met O 2b Technical Note No. 25.
Temperton, C.	1979	Fast Fourier Transforms on Cray-1 E.C.M.W.F. Internal Report 21.
Temperton, C.	1982	Fast Real Fourier Transforms on the Cyber 205. Met O 11 Technical Note No. 155.
Williamson, D.L.	1976	Linear Stability of Finite-Difference Approximations on a Uniform Latitude-Longitude Grid with Fourier Filtering. Monthly Weather Review Vol. 104 No. 1 pp 31-41.

APPENDIX I Alphabetical List Of Subroutines and Functions

ADVISE	A stub routine. In a paging model, ADVISE must be replaced by a routine which informs the operating system that an area of DOMAIN should be paged in or out.
ASKOP	A logical function which returns .TRUE. if the operator has asked the model to stop.
BUFIN	Reads a field or a header from a model dump or boundary data.
BUFOUT	Writes a field or a header to a model dump.
CHOP	The fourier truncation routine. Truncates one latitude of the grid.
CLCNSA	Called by DYNNSA to calculate increments due to north-south advection.
CLCNSW	Called by DYNNSW to calculate vertical velocity and north-south differences of wind and temperature.
CLCNSZ	Called by DYNNSZ to calculate the north-south derivative of geopotential.
CPOLE	Rotates data in one or more rows by 180°.
CPYDOM	Overwrites the fields at the update time level with the values of the fields at the current time level. CPYDOM is called before a forward timestep is performed.
CPYJCL	Copies the tidy-up JCL to the file containing the restart JCL. Called at the end of an integration to stop the model from restarting.
CRSPOL	Rotates a "slice" of temperature and wind fields by 180° and changes the sign of the wind fields. Used before cross-polar differencing and filtering.

DATIM	Returns the date and time of the model run.
DATSEC	A double precision function which converts a date to the number of seconds since midnight on 1st January year zero.
DMPTIM	A logical function which returns .TRUE. if the integration has reached a point where an output dump should been produced.
DYNAMIC	The dynamics routine. Calculates increments for one latitude of the grid.
DYNCOR	Calculates increments due to the Coriolis effect.
DYNEWA	Calculates increments due to east-west advection.
DYNNSA	Calls CRSPOL and CLCNSA to calculate increments due to north-south advection.
DYNNSW	Calls CRSPOL and CLCNSW to calculate vertical velocity and north-south differences of temperature and wind.
DYNNSZ	Calls CPOLE and CLCNSZ to calculate the north-south derivative of geopotential.
DYNVAD	Calculates increments due to vertical advection.
D1GTD2	A logical function which compares two dates.
ENDRUN	Called at the end of an integration, writes the final dump and calls CPYJCL before stopping.
FFT	The fast fourier transform routine.
FILTEW	Applies a Shapiro filter to one latitude of the model.
FILTNS	Applies a Shapiro filter in the north-south direction by alternate passes of north-to-south and south-to-north operators.
FLTNTS	Called by FILTNS to apply a north-to-south operator.
FLTSTN	Called by FILTNS to apply a south-to-north operator.

FNDUMP	Locates a dump or boundary field in a file by its date.
FULFIL	A logical function which returns .TRUE. when enough output dumps have been produced to fill the output file.
INCDAT	Increments a date.
INIPAG	Part of the paging scheme. Sets up the array and variables which keep track of the parts of DOMAIN which are in memory or paged out to disk.
MODEL	The main program. Opens the files used by the model, reads constants into IPARMS and RPARMS by calling RDPARM, reads start and stop dates etc. by calling RDCARD, and passes control to STRTUP.
MODHED	Creates a header for the output dumps.
NEWTON	The Newtonian cooling scheme.
OPSTOP	Called when the operator has asked the model to stop. Writes a dump to the restart dump file and then stops the model.
PAGOUT	Pages out a latitude of the grid by calling ADVISE.
PGNEXT	Pages in an out-of-memory latitude of the grid by calling ADVISE.
PRNPRM	Prints some of the constants from IPARMS and RPARMS.
QNORM	Called by INCDAT to normalise a date by e.g. converting 0/1/13/1979 to 0/1/1/1980.
RAYLEY	The Rayleigh friction routine.
RDBDY	Reads a boundary field.
RDCARD	Reads the start and stop dates etc.
RDDUMP	Reads an initial data dump.
RDINIT	Reads the initial data by calling FNDUMP to locate it and RDDUMP to read it.

RDPARM	Reads constants into IPARMS and RPARMS.
READ	Called by BUFIN to read a field or a header from a file.
RESEND	Restores the values of the first and last elements of a row of data.
RESTRT	Called when the output file is full. Writes a dump to the restart dump file and then stops the model.
SETCHP	Calculates coefficients for the fourier truncation routine CHOP.
SETDYN	Calculates coefficients for the dynamics routines.
SETNWT	Calculates coefficients for the Newtonian cooling routine NEWTON.
SETRAY	Calculates coefficients for the Rayleigh friction routine RAYLEY.
STOP	Tests if the operator has asked the model to stop.
STRTUP	Performs some of the processing required before the main time loop begins. Reads initial data by calling RDINIT. Creates a header for the output dumps by calling MODHED, initialises the boundary interpolation process by calling UPDBDY, and then passes control to TMLOOP.
TIMSTP	Performs one timestep. TIMSTP contains the loop over latitude and calls the paging routines INIPAG, PAGOUT and PGNEXT to control paging.
TMLOOP	The time loop routine. Performs timesteps by calling TIMSTP, writes changes to the output file by calling WRDUMP, updates the boundary field each timestep by calling UPDBDY, and stops the integration by calling ENDRUN or OPSTOP.

TSTBDY	Checks the header of a boundary data dump for the correct grid size and pressure.
TSTINI	Checks the header of the initial data dump for the correct grid size and boundary pressures.
TSTPRM	Tests the content of IPARMS and RPARMS for sensible values.
UPDBDY	The first call initialises the routine and causes it to read a boundary data field. Subsequent calls interpolate the boundary forward in time, reading new data when necessary.
UPDLAT	Called by TIMSTP to timestep one latitude of the model.
UPDTMF	Adds increments to the "update" fields and performs time filtering.
WRDUMP	Writes a dump to the output file unless it is full in which case it stops the model by calling RESTRT.
WRHED	Called by BUFOUT to write a header.
WRIDMP	Writes a dump to a file.

Appendix II Dump Format Data

This appendix is included for completeness only. Dump format data should not be manipulated by the user except via the model's I/O routines BUFIN and BUFOUT. The user, therefore, does not need to know the method by which these routines store data in a file.

BUFIN and BUFOUT use FORTRAN unformatted I/O to transfer data. Each call to BUFOUT results in the writing of two records. The first record of the pair consists of the eight character name and the integer number of elements given in the argument list, except if "*TOP " or "*NEXT " is specified in which case the record consists of the pair "*END " and 1024. The second record contains the data from the array given in the argument list. The BUFIN routine uses the first record of each pair to identify fields of data by name and to locate the beginning and ends of dumps.

Random access to the fields of a dump is provided by an internal table in the BUFIN routine containing the names of fields between the start of the dump and the position of the reading device. This allows BUFIN to reposition the device using the BACKSPACE statement to access fields whose names are in the table. If a name is not found in the table, the reading device is moved down the dump until the corresponding field is located.

Appendix III Paging Scheme Performance

Let 'p' represent the time required for the computer to move a single page into real memory. It is assumed that this process occurs simultaneously with the transfer of another page out of memory and with CPU activity in a third region of memory. Let 'n' be the number of pages per latitude of the grid, and let 'c' be the time required by the CPU for the calculations at one latitude of the grid.

Given a real memory of 'm' pages, there is room for (m/n) latitudes of the model to reside in real memory. A model with 'l' latitudes in its grid will have (l-m/n) latitudes which require paging (assuming $l > m/n$). By scattering these areas at intervals throughout the model, the paging scheme allows the I/O for each of the paged latitudes to be concurrent with the calculations for $l/(l - m/n)$ latitudes. The model is therefore CPU bound provided

$$np\,(l - m/n) < cl$$

ie $$l < mp/(np - c)$$

As a concrete example, consider a 2° x 2° x 32 level model using half precision arithmetic on a 2 Mword Cyber 205. Projecting the performance of the 5° x 5° x 32 level model, which requires approximately .2s per time step, we may calculate 'c' to be .014s. The computer is characterized by p = .05 s/page, m = 800 pages and a page size of 4096 half words. The model has ninety latitudes, each requiring 9 pages (i.e. l = 90, n = 9). Substituting these numbers into the inequality above, we have:

$mp/(np - c) = 91.7$ whereas $l = 90$

The model is therefore CPU bound so the time required for a single timestep is given by

$cl = 1.26$ s

Fig. 2 illustrates the performance of the scheme in comparison with a simple paging scheme which pages every latitude of the model each time step. Also shown is the performance of the model without a paging scheme. Note that the scheme adopted allows the model to remain CPU bound at higher resolutions than the simpler alternative, and results in a saving of np(m/n) seconds per timestep, equivalent to the time required to page the entire physical memory.

Appendix IV Notation

a	radius of the earth.
p	pressure
$p_o = 1$ Pa	reference pressure
$s = \log (p/p_o)$	vertical co-ordinate
t	time
u	eastwards component of wind
v	northwards " " "
$f = 2\Omega\sin\phi$	Coriolis parameter
$\dot{s} = Ds/Dt$	"vertical velocity"
F_λ, F_ϕ	forcing of wind due to e.g. gravity wave induced drag.
Q	diabatic heating
T	temperature
C_p	Specific heat of air at constant pressure.
λ	longitude
ϕ	latitude
Φ	geopotential

Finite difference operators

Let $X = X(x,\ldots)$, $Y = Y(x,\ldots)$

Then
$$\delta_{nx}X = [X(x + n\Delta x/2,\ldots) - X(x - n\Delta x/2,\ldots)]/n\Delta x$$
$$\overline{X}^{nx} = [X(x + n\Delta x/2,\ldots) + X(x - n\Delta x/2,\ldots)]/2$$
$$\widetilde{XY}^{x} = [X(x + \Delta x/2,\ldots)\, Y(x - \Delta x/2,\ldots) + X(x - \Delta x/2,\ldots)\, Y(x + \Delta x/2,\ldots)]/2$$
$$\delta_x X = \delta_{1x}X$$
$$\overline{X}^{x} = \overline{X}^{1x}$$

Appendix V Finite Difference Accuracy Near the Poles

An unfortunate feature of "box" type finite differences when applied to the primitive equations on a sphere, is the form which the meridional gradient of geopotential must take in order to conserve energy. Using second order accurate differences, this term is:

$$(a\cos\phi)^{-1}[\delta_\phi(\bar{\Phi}^\phi\cos\phi) - \Phi\delta_\phi\cos\phi]$$

ie

$$(2a\Delta\phi\cos\phi)^{-1}\ \{\ [\Phi(\phi + \Delta\phi,...) - \Phi(\phi,...)]\cos(\phi + \Delta\phi/2)$$

$$- [\Phi(\phi - \Delta\phi,...) - \Phi(\phi,...)]\cos(\phi - \Delta\phi/2)\ \}$$

$$= (2a\Delta\phi)^{-1}\ \{\ \cos(\Delta\phi/2)[\Phi(\phi + \Delta\phi,...) - \Phi(\phi - \Delta\phi,...)]$$

$$- [\Phi(\phi + \Delta\phi,...) + \Phi(\phi - \Delta\phi,...) - 2\Phi(\phi)]\tan\phi\sin(\Delta\phi/2)\ \}$$

Expanding the trigometric functions in Taylor series about zero, and the terms in square brackets about $\Phi(\phi)$ gives:

$$\frac{1}{a}\frac{\partial\Phi}{\partial\phi} + A\,\Delta\phi^2 + B\Delta\phi^4 + \tan\phi(C\Delta\phi^2 + D\Delta\phi^4) + O(\Delta\phi^6)$$

where A, B, C, D are $O(\Delta\phi^0)$.

The approximation is thus generally of second order accuracy. However, near the poles $\tan\phi \sim \Delta\phi^{-1}$ and the approximation is only first order accurate (Kalney-Rivas 1978).

Extension of this analysis to the fourth order equations of the stratosphere-mesosphere model is straightforward if the equations are regarded as a linear combination of second order differences on the model's grid and a grid with twice the horizontal grid spacing.

The corresponding term of the model's equation is:

$$\frac{4}{3}(a\cos\phi)^{-1}[\delta_\phi(\bar{\Phi}^\phi\cos\phi) - \Phi\delta_\phi\cos\phi]$$

$$-\frac{1}{3}(a\cos\phi)^{-1}[\delta_{2\phi}(\bar{\Phi}^{2\phi}\cos\phi) - \Phi\delta_{2\phi}\cos\phi]$$

$$= \frac{4}{3}\left[\frac{1}{a}\frac{\partial\phi}{\partial\phi} + A\Delta\phi^2 + B\Delta\phi^4 + \tan\phi(C\Delta\phi^2 + D\Delta\phi^4)\right]$$

$$- \frac{1}{3}\left[\frac{1}{a}\frac{\partial\phi}{\partial\phi} + 4A\Delta\phi^2 + 16B\Delta\phi^4 + \tan\phi(4C\Delta\phi^2 + 16D\Delta\phi^4)\right]$$

$$+ O(\Delta\phi^6)$$

$$= \frac{1}{a}\frac{\partial\phi}{\partial\phi} - 4B\Delta\phi^4 - \tan\phi(4D\Delta\phi^4) + O(\Delta\phi^6)$$

The approximation is thus fourth order accurate away from the poles and becomes third order accurate as $\tan\phi \to \Delta\phi^{-1}$ near the poles.

Appendix VI List of Constants in File PARAMS

Arrays IPARMS and RPARMS are filled at run time with 256 constants each by the following FORTRAN statements.

```
READ (16,'(8I10)'  )IPARMS
READ (16,'(8G10.3)')RPARMS
```

Only a few elements of these arrays are used by the base-code model. Later versions of the model may reserve other elements for specific use. The model protects itself against using an inappropriate set of constants by testing the version number in the first element of IPARMS. It is recommended that unused elements be set to zero or to some obvious value such as -999. The following constants are required.

IPARMS(1)	-----	The model version number as specified in the first parameter statement of the main problem.
IPARMS(2)	-----	The number of longitude points in the grid (=72 for 5 degree resolution).
IPARMS(3)	-----	The number of latitude points in the grid (=36 for 5 degree resolution).
IPARMS(4)	-----	The number of model levels (excluding the lower boundary).
IPARMS(5)	-----	The model's time-step in seconds.
IPARMS(33)	-----	The number of latitudes at each pole which require Fourier truncation.

IPARMS(34) onwards	-----	The lowest zonal wave number to be removed at the polar rows (in pole to equator order).
RPARMS(1)	-----	The pressure in Pascals of the lower boundary.
RPARMS(2)	-----	The pressure in Pascals of the top level of the model.
RPARMS(17)	-----	Time filter coefficient ν
RPARMS(33)	-----	g (mks units)
RPARMS(34)	-----	R (mks units)
RPARMS(35)	-----	κ (mks units)
RPARMS(36)	-----	radius of the earth (metres)
RPARMS(37)	-----	angular speed of the earth (radians/s)
RPARMS(38)	-----	scale height (metres)
RPARMS(39)	-----	surface pressure (Pascals)

Appendix VII Transporting Data between computers

The model has two methods for transporting model dumps and other data written by the output routine BUFOUT. The first method is to convert the data to "transport format". This is a portable but inefficient data format which is used to allow easy transfer of data between computer installations. The second method is to convert the data directly to the representation used by a particular computer. This method is used to transfer data between "front end" and "back end" machines.

Transport Format

Transport format data is stored as ASCII characters in fixed length 80 byte records which are written and read using FORTRAN formatted I/O. Each field of data is preceded by a record containing the name and number of elements in the field. The elements are represented in character form using E12.6E2 format. However, before storing the numbers, the redundant decimal point and 'E' are removed allowing ten elements to be stored in each record. It is hoped that the six figure accuracy of these numbers is a reasonable compromise between the conflicting requirements of concision and precision.

Two FORTRAN-77 programs exist to convert data from transport format to the more efficient representation used by the model's I/O routines, and vice versa. They are listed on the microfiche which accompanies this note. The programs make no assumption about the word length or internal representation of numbers used by the computer on which they run, and are therefore portable. However, since the header record may vary from machine to machine, slight modification to the programs may be required to

rearrange header records before writing or after reading them. This is an unfortunate consequence of ignoring FORTRAN data types and storing character, integer and real data in a single array.

Direct Conversion

Direct conversion of data to the representation used by another machine is both more efficient and more accurate than using transport format but requires programs which are specific to a pair of computers. At present this method of transporting data has been implemented only for transfer between the IBM and Cyber 205 computers of the Met Office's COSMOS installation. The conversion programs are listed on the microfiche which accompanies this note. Numbers and characters are converted using routines written at the Met. Office for this purpose (Foreman 1981). The internal file structure is converted by the software which controls the link between the two machines.

Program Structure

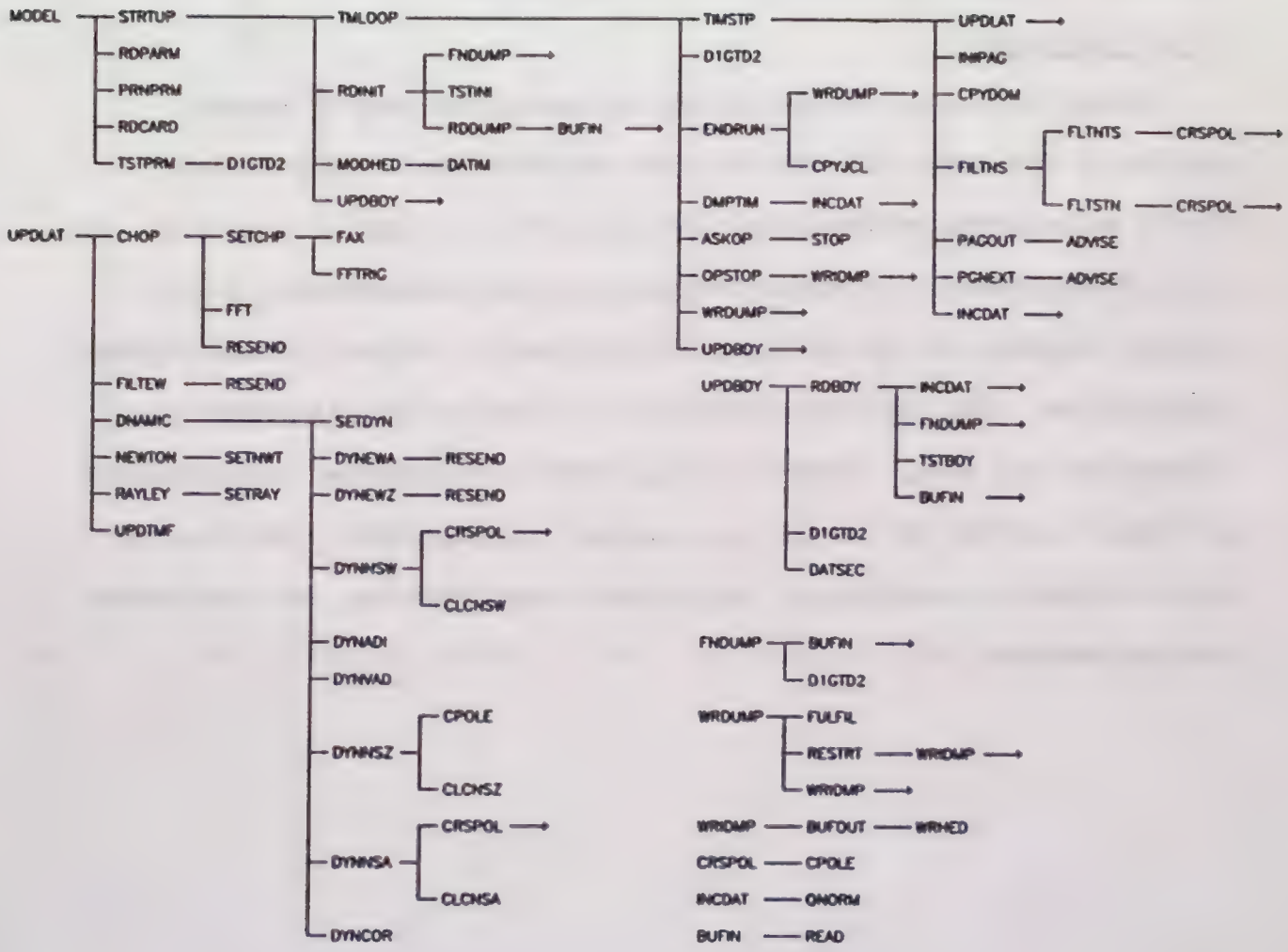

Fig.1

Paging Scheme Performance

(not to scale)

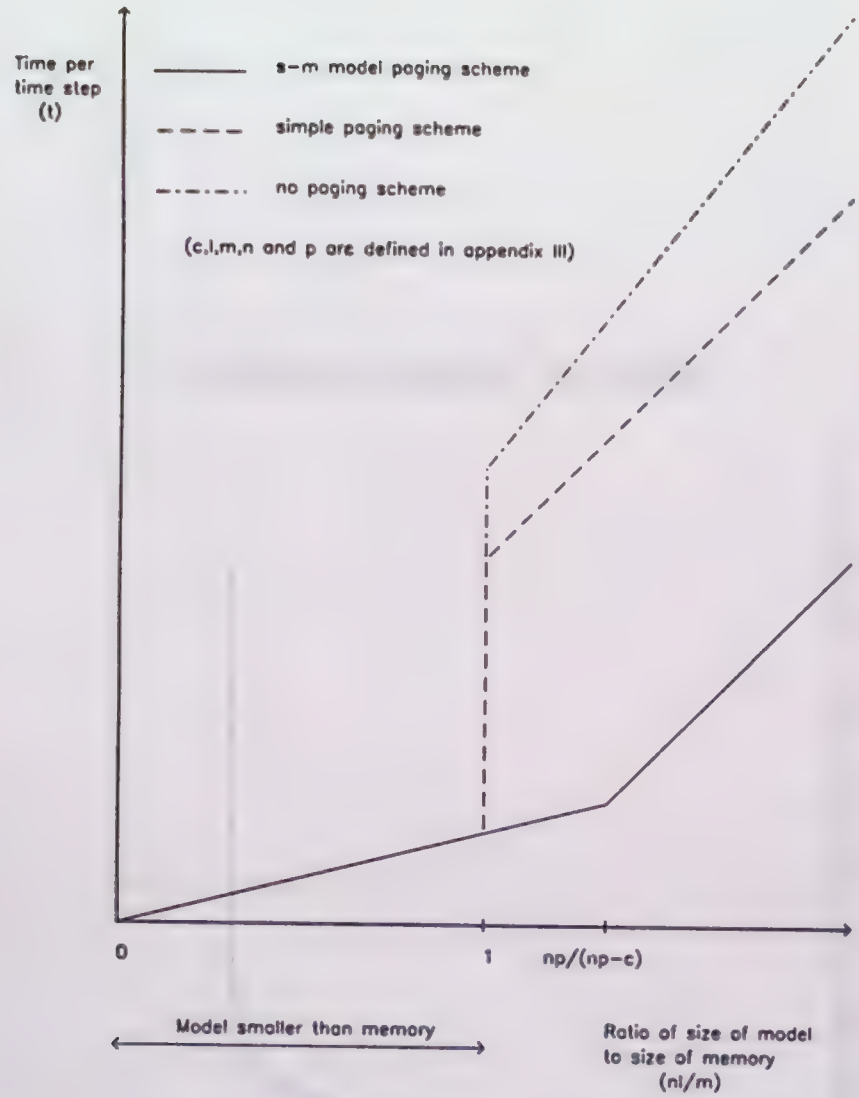

Fig.2

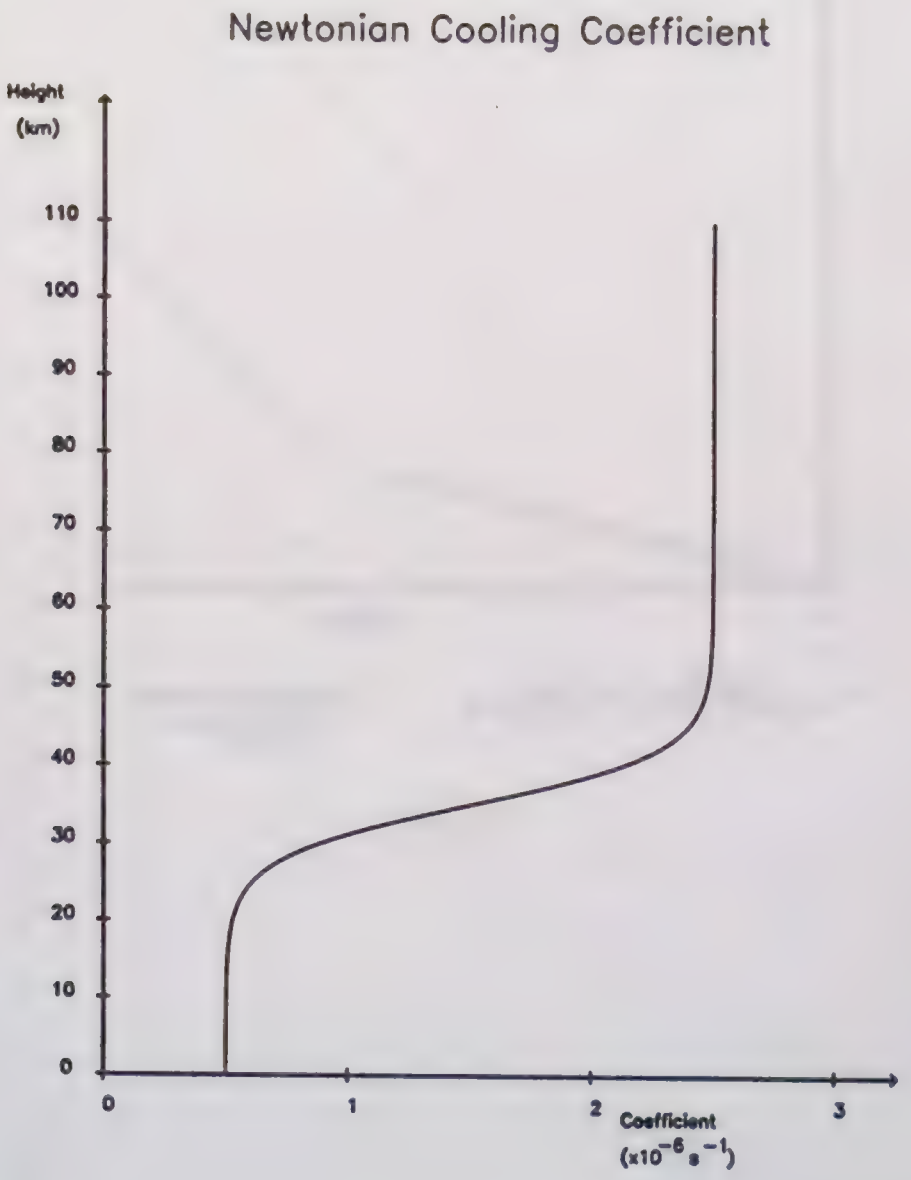
Newtonian Cooling Coefficient
Height
(km)
110
100
90
80
70
60
50
40
30
20
10
0
0
1
2
3
Coefficient
(x10⁻⁶ s⁻¹)

Fig.3

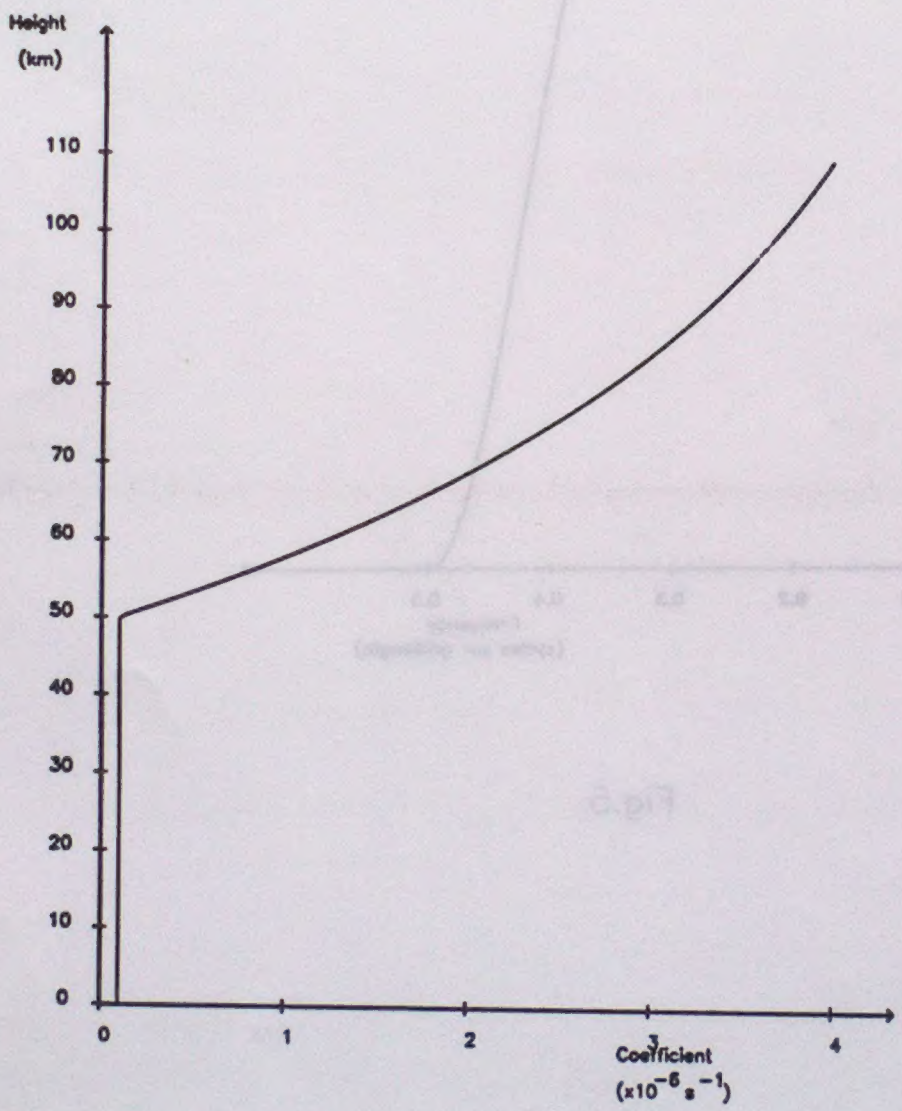
Rayleigh Friction Coefficient
Height
(km)
110
100
90
80
70
60
50
40
30
20
10
0
0
1
2
3
4
Coefficient
(x10⁻⁶ s⁻¹)

Fig.4

Eighth Order Shapiro Filter Response

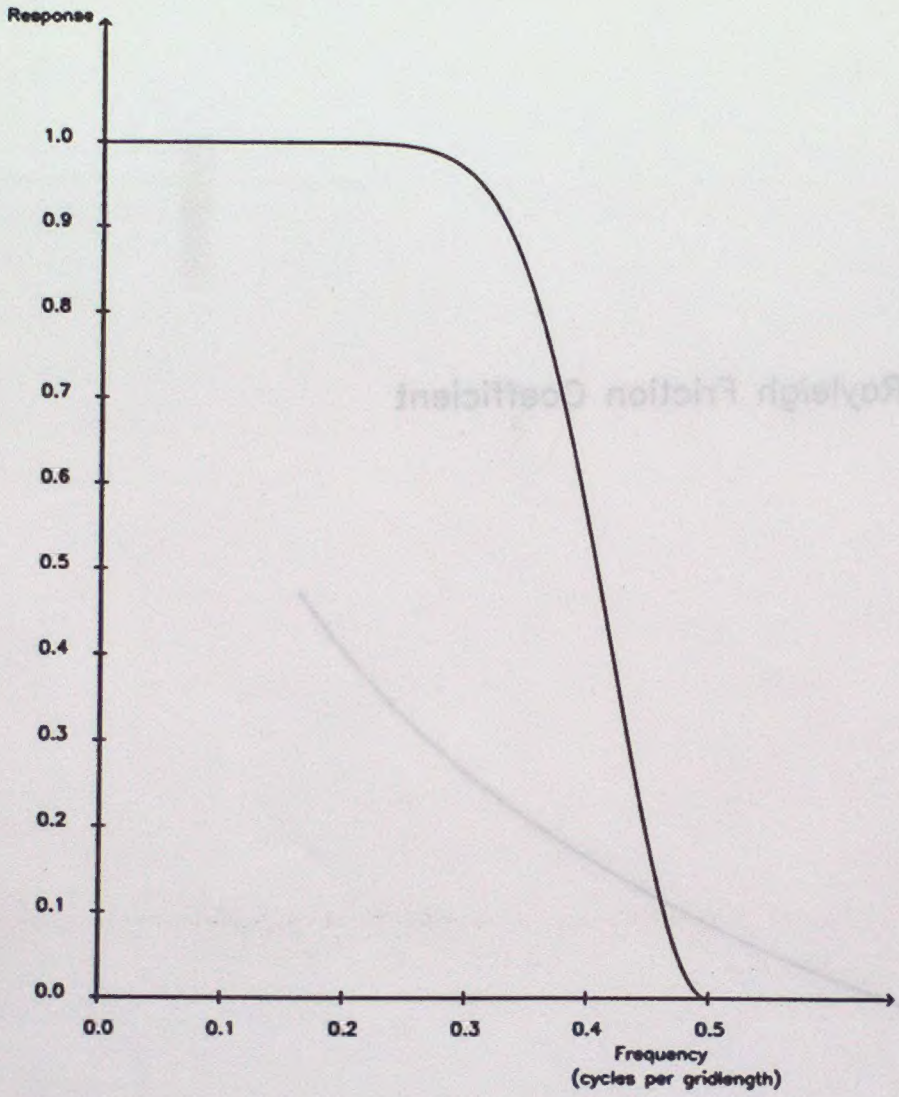

Fig.5

DYNAMICAL CLIMATOLOGY TECHNICAL NOTES

DCTN 1	June 1984	A scheme for incorporating the thermodynamic sea-ice model into a coupled ocean-atmosphere model. C Gordon and M Bottomley
DCTN 2	Sept 1984	The parametrization of the upper ocean mixed layer in coupled ocean/atmosphere models. C Gordon and M Bottomley
DCTN 3	Sept 1984	The response of a simple thermodynamic sea-ice model to GCM forcing appropriate for perennial sea-ice. M Bottomley
DCTN 4	Sept 1984	Potential vorticity in the stratosphere derived using data from satellites. S A Clough, N S Grahame and A O'Neill
DCTN 5	Sept 1984	Simulation of the seasonal cycle of SST using globally specified climatological data to force simple ocean models. N S Grahame
DCTN 6	Sept 1984	Development of an improved GCM radiation scheme. A Slingo and R C Wilderspin
DCTN 7	Sept 1984	On the specification of surface fluxes in coupled atmosphere/ocean general circulation models. J F B Mitchell, C A Wilson and C M Price
DCTN 8	Sept 1984	A survey of possible repercussions of the Soviet river reversal programme on Arctic sea-ice. H Cattle
DCTN 9	Oct 1984	A study of a strong stratospheric warming during January 1982 T D A Fairlie
DCTN 10	Nov 1984	An intercomparison of the Met O 11 M45 model and the Met O 20 11-level model on the Cyber 205 J F Dyson
DCTN 11	Nov 1984	Simulation of observed time series of temperature and precipitation over eastern England. D M Reed
DCTN 12	Nov 1984	Forecasts and analyses of the onset of the Southwest monsoon. R Kershaw
DCTN 13	Nov 1984	On modelling the effects of CO_2 on climate. J F B Mitchell
DCTN 14	Dec 1984	Development of a revised longwave radiation scheme for an atmospheric general circulation model. A Slingo and R C Wilderspin
DCTN 15	Jan 1985	Rossby wavetrains in the stratosphere forced by localised disturbances in the troposphere. C J Marks (Oxford), A O'Neill and V D Pope

DYNAMICAL CLIMATOLOGY TECHNICAL NOTES

DCTN 16	Jan 1985	Observing system experiments using the Meteorological Office's 15-level model. R A Bromley
DCTN 17	Feb 1985	Atmospheric general circulation simulation of the climate of the Mediterranean region. H Cattle
DCTN 18	Feb 1985	Diurnal variation and cloud in a general circulation model. C A Wilson and J F B Mitchell
DCTN 19	Feb 1985	The sensitivity of the Saharan region in a general circulation model. W M Cunnington and P R Rowntree
DCTN 20	Mar 1985	The impact of aircraft observations on analyses and forecasts from the operational 15-level model. B R Barwell
DCTN 21	Mar 1985	Non-linear effects on heat-induced circulation of the tropical atmosphere. A E Gill and P J Phlips
DCTN 22	Apr 1985	Simulation of cloudiness with the U.K. Meteorological Office 11-layer AGCM. A Slingo and R C Wilderspin
DCTN 23	Apr 1985	An integrated approach to the representation of cloud processes. R N B Smith
DCTN 24	May 1985	Nuclear Winter: Calculations of the shortwave radiative effects of soot aerosols. A Slingo and P Goldsmith
DCTN 25	Jul 1985	Sea ice and the Antarctic winter circulation: a numerical experiment. J F B Mitchell and T S Hills
DCTN 26	Jul 1985	A Budyko-Sellers model for study of ocean-atmosphere interaction. W J Ingram
DCTN 27	Aug 1985	The effect of a shoaling thermocline on equatorially-trapped Kelvin waves. A E Gill and B A King
DCTN 28	Sept 1985	CO_2 climate sensitivity and model dependence of results. J F B Mitchell, C A Wilson and W M Cunnington
DCTN 29	Sept 1985	Handbook of the Meteorological Office 11-layer atmospheric general circulation model. Volume 1 : Model Description Edited by A Slingo
DCTN 30	Sept 1985	Impact of land surface variations on African rainfall in general circulation models. P R Rowntree, M F Wilson and A B Sangster

DCTN 31 Nov 1985 The annual cycle of sea level in the eastern tropical Pacific.
G R Bigg and A E Gill

DCTN 32 Nov 1985 An analytic model of the heat-induced tropical circulation in the presence of a mean wind.
P J Phlips and A E Gill

DCTN 33 Nov 1985 The effect of resolution and diffusion on the simulated climate.
J F Dyson

DCTN 34 Jan 1986 Rossby adjustment over a step.
M K Davey, A E Gill, E R Johnson and P F Linden

DCTN 35 Jan 1986 Issues concerning the evaluation of effective surface roughness of heterogeneous terrain.
D J Carson

DCTN 36 Mar 1986 Climate sensitivity and past climates : evidence from numerical studies
J F B Mitchell

DCTN 37 Mar 1986 Parametrizations of land-surface processes in Meteorological Office numerical weather prediction and climate models
D J Carson

DCTN 38 Apr 1986 Modelling of land-surface processes and their influence on European Climate
D A Warrilow, A B Sangster and A Slingo

DCTN 39 Apr 1986 Simulation of climatic change due to increased CO_2
J F B Mitchell

DCTN 40 May 1986 On the interpretation of the Eliassen-Palm flux divergence.
Part I : Theory
D G Andrews

DCTN 41 June 1986 Experiments on tropical circulation with a simple moist model.
M K Davey and A E Gill

DCTN 42 Jun 1986 Parameter regimes for studying isolated eddies.
P D Killworth and M K Davey

DCTN 43 Jun 1986 A comparison of the impact of an envelope orography and of a parametrization of orographic gravity-wave drag on model simulations.
A Slingo and D W Pearson

DCTN 44 Jul 1986 The effect of a sea-surface temperature anomaly on a prediction of the onset of the south-west monsoon over India.
R Kershaw

DCTN 45 Jul 1986 An introduction to the parametrization of land-surface processes.
D J Carson

DYNAMICAL CLIMATOLOGY TECHNICAL NOTES

DCTN 46 Sept 1986 Lateral boundaries and the heat-induced flow in the tropical atmosphere.
P J Phlips and A E Gill

DCTN 47 Nov 1986 Simulation of the diurnal cycle of outgoing longwave radiation with an atmospheric GCM.
A Slingo, R C Wilderspin and S J Brentnall

DCTN 48 Feb 1987 A 2 x CO_2 climate sensitivity experiment with a global climate model including a simple ocean
C A Wilson and J F B Mitchell

DCTN 49 Feb 1987 Sensitivity of the Met O 20 11-layer model to increased land surface roughness length
C C Heasman

DCTN 50 Feb 1987 The influence of atmospheric waves on the general circulation of the middle atmosphere
D G Andrews

DCTN 51 Feb 1987 Transport mechanisms in the middle atmosphere: an introductory survey
D G Andrews